Theoretical and applied aspects of the self-organizing maps

Marie Cottrell, Madalina Olteanu, Fabrice Rossi, Nathalie Vialaneix

Theoretical and Applied Aspects of the Self-Organizing Maps

Marie Cottrell[1], Madalina Olteanu[1], Fabrice Rossi[1], and Nathalie Villa-Vialaneix[2]

[1] SAMM - Université Paris 1 Panthéon-Sorbonne
90, rue de Tolbiac, 75013 Paris, France
marie.cottrell,madalina.olteanu,fabrice.rossi@univ-paris1.fr
[2] INRA, UR 0875 MIAT
BP 52627, F-31326 Castanet Tolosan, France
nathalie.villa@toulouse.inra.fr

Abstract. The Self-Organizing Map (SOM) is widely used, easy to implement, has nice properties for data mining by providing both clustering and visual representation. It acts as an extension of the k-means algorithm that preserves as much as possible the topological structure of the data. However, since its conception, the mathematical study of the SOM remains difficult and has be done only in very special cases. In WSOM 2005, Jean-Claude Fort presented the state of the art, the main remaining difficulties and the mathematical tools that can be used to obtain theoretical results on the SOM outcomes. These tools are mainly Markov chains, the theory of Ordinary Differential Equations, the theory of stability, etc. This article presents theoretical advances made since then. In addition, it reviews some of the many SOM algorithm variants which were defined to overcome the theoretical difficulties and/or adapt the algorithm to the processing of complex data such as time series, missing values in the data, nominal data, textual data, etc.

Keywords: SOM, batch SOM, Relational SOM, Stability of SOM

1 Brief history of the SOM

Since its introduction by T. Kohonen in his seminal 1982 articles (Kohonen [1982a,b]), the self-organizing map (SOM) algorithm has encountered a very large success. This is due to its very simple definition, to the easiness of its practical development, to its clustering properties as well as its visualization ability. SOM appears to be a generalization of basic clustering algorithms and at the same time, provides nice visualization of multidimensional data.

The basic version of SOM is an on-line stochastic process which has been inspired by neuro-biological learning paradigms. Such paradigms had previously been used to model some sensory or cognitive processes where the learning is directed by the experience and the external inputs without supervision. For example, Ritter and Schulten [1986] illustrate the somatosensory mapping property of

SOM. However, quickly in the eighties, SOM was not restricted to neuro-biology modeling and has been used in a huge number of applications (see e.g. Kaski et al. [1998b], Oja et al. [2003] for surveys), in very diverse fields as economy, sociology, text mining, process monitoring, etc.

Since then, several extensions of the algorithms have been proposed. For instance, for users who are not familiar with stochastic processes or for industrial applications, the variability of the equilibrium state was seen as a drawback because the learnt map is not always the same from one run to another. To address this issue, T. Kohonen introduces the batch SOM, (Kohonen [1995, 1999]) which is deterministic and thus leads 3 to reproducible results (for a given initialization). Also, the initial SOM (on-line or batch versions) was designed for real-valued multidimensional data, and it has been necessary to adapt its definition in order to deal with complex non vectorial data such as categorical data, abstract data, documents, similarity or dissimilarity indexes, as introduced in Kohonen [1985], Kaski et al. [1998a], Kohohen and Somervuo [1998]. One can find in Kohonen [1989, 1995, 2001, 2013, 2014] extensive lists of references related to SOM. At this moment more than 10 000 papers have been published on SOM or using SOM.

In this paper, we review a large selection of the numerous variants of the SOM. One of the main focus of this survey is the question of *convergence* of the SOM algorithms, viewed as stochastic processes. This departs significantly from the classical learning theory setting. In this setting, exemplified by the pioneering results of Pollard et al. [1981], one generally assumes given an optimization problem whose solution is interesting: for instance, an optimal solution of the quantization problem associated to the k-means quality criterion. The optimization problem is studied with two points of view: the true problem which involves a mathematical expectation with respect to the (unknown) data distribution and its empirical counterpart where the expectation is approximated by an average on a finite sample. Then the question of convergence (or consistency) is whether the solution obtained on the finite sample converges to the true solution that would be obtained by solving the true problem.

We focus on a quite different problem. A specific stochastic algorithm such as the SOM one defines a series of intermediate configurations (or solutions). Does the series converge to something interesting? More precisely, as the algorithm maps the inputs (the data) to an output (the prototypes and their array), one can take this output as the result of the learning process and may ask the following questions, among others:

- How to be sure that the learning is over?
- Do the prototypes extract a pertinent information from the data set?
- Are the results stables?
- Are the prototypes well organized?

In fact, many of these questions are without a complete answer, but in the following, we review parts of the questions for which theoretical results are known and summarize the main remaining difficulties. Section 2 is devoted to the definition of SOM for numerical data and to the presentation of the general methods

used for studying the algorithm. Some theoretical results are described in the next sections: Section 3 explains the one dimensional case while in Section 4, the results available for the multidimensional case are presented. The batch SOM is studied in Section 5. In Section 6, we present the variants proposed by Heskes to get an energy function associated to the algorithm. Section 7 is dedicated to non numerical data. Finally, in Section 8, we focus on the use of the stochasticity of SOM to improve the interpretation of the maps. The article ends with a very short and provisional conclusion.

2 SOM for numerical data

Originally, (in Kohonen [1982a] and Kohonen [1982b]), the SOM algorithm was defined for vector numerical data which belong to a subset $\mathcal{X}$ of an Euclidean space (typically $\mathbb{R}^p$). Many results in this paper additionally require that the subset is bounded and convex. There are two different settings from the theoretical point of view:

- *continuous setting*: the input space $\mathcal{X}$ can be modeled by a probability distribution defined by a density function f;
- *discrete setting*: the data space $\mathcal{X}$ comprises N data points $x_1, \ldots, x_N$ in $\mathbb{R}^p$ (In this paper, by *discrete setting*, we mean a *finite* subset of the input space).

The theoretical properties are not exactly the same in both cases, so we shall later have to separate these two settings.

2.1 Classical on-line SOM, continuous or discrete setting

In this section, let us consider that $\mathcal{X} \subset \mathbb{R}^p$ (*continuous or discrete setting*).

First we specify a regular lattice of K units (generally in a one- or two-dimensional array). Then on the set $\mathcal{K} = \{1, \ldots, K\}$, a neighborhood structure is induced by a neighborhood function h defined on $\mathcal{K} \times \mathcal{K}$. This function can be time dependent and, in this case, it will be denoted by $h(t)$. Usually, h is symmetrical and depends only on the distance between units k and l on the lattice (denoted by $\text{dist}(k, l)$ in the following)). It is common to set $h_{kk} = 1$ and to have k_{kl} decrease with increasing distance between k and l. A very common choice is the step function, with value 1 if the distance between k and l is less than a specific radius (this radius can decrease with time), and 0 otherwise. Another very classical choice is

$$h_{kl}(t) = exp\left(-\frac{\text{dist}^2(k,l)}{2\sigma^2(t)}\right),$$

where $\sigma^2(t)$ can decrease over time to reduce the intensity and the scope of the neighborhood relations.

A prototype $m_k \in \mathbb{R}^p$ is attached to each unit k of the lattice. Prototypes are also called models, weight vectors, code-vectors, codebook vectors, centroids,

etc. The goal of the SOM algorithm is to update these prototypes according to the presentation of the inputs in such a way that they represent the input space as accurately as possible (in a quantization point of view) while preserving the topology of the data by matching the regular lattice with the data structure. For each prototype m_k, the set of inputs closer to m_k than to any other one defines a cluster (also called a Voronoï cell) in the input space, denoted by C_k, and the neighborhood structure on the lattice induces a neighborhood structure on the clusters. In other words, after running the SOM process, close inputs should belong to the same cluster (as in any clustering algorithm) or to neighbor clusters.

From any initial values of the prototypes, $(m_1(0), \ldots, m_K(0))$, the SOM algorithm iterates the following steps:

1. At time t, if $m(t) = (m_1(t), \ldots, m_K(t))$ denotes the current state of the prototypes, a data point x is drawn according to the density f in $\mathcal{X}$ (*continuous setting*) or at random in the finite set $\mathcal{X}$ (*discrete setting*).
2. Then $c^t(x) \in \{1, \ldots, K\}$ is determined as the index of the *best matching unit*, that is
$$c^t(x) = \arg \min_{k \in \{1,\ldots,K\}} \|x - m_k(t)\|^2, \tag{1}$$
3. Finally, all prototypes are updated via
$$m_k(t+1) = m_k(t) + \epsilon(t) h_{kc^t(x)}(t)(x - m_k(t)), \tag{2}$$
where $\epsilon(t)$ is a learning rate (positive, less than 1, constant or decreasing).

Although this algorithm is very easy to define and to use, its main theoretical properties remain without complete proofs. Only some partial results are available, despite a large amount of works and empirical evidences. More precisely, $(m_k(t))_{k=1,\ldots,K}$ are K stochastic processes in $\mathbb{R}^p$ and when the number t of iterations of the algorithm grows, $m_k(t)$ could have different behaviors: oscillation, explosion to infinity, convergence in distribution to an equilibrium process, convergence in distribution or almost sure to a finite set of points in $\mathbb{R}^p$, etc.

This is the type of convergence that we will discuss in the sequel. In particular, the following questions will be addressed:

- Is the algorithm convergent in distribution or almost surely, when t tends to $+\infty$?
- What happens when ϵ is constant? when it decreases?
- If a limit state exists, is it stable?
- How to characterize the organization?

One can find in Cottrell et al. [1998] and Fort [2006] a summary of the main rigorous results with most references as well as the open problems without solutions until now.

2.2 Mathematical tools related to the convergence of stochastic processes

The main methods that have been used to analyze the SOM convergence are summarized below.

- *The Markov Chain theory* for constant learning rate and neighboring function, which is useful to study the convergence and the limit states. If the algorithm converges in distribution, this limit distribution has to be an invariant measure for the Markov Chain. If it is possible to prove some strong organization, it has to be associated to an absorbing class;

- The *Ordinary Differential Equation method* (ODE), which is a classical method to study the stochastic processes.
  If we write down the equation (2) for each $k \in \mathcal{K}$ in a vector form, we get

$$m(t+1) = m(t) - \epsilon(t)\Phi(x, m(t)), \tag{3}$$

  where Φ is a stochastic term. To study the behavior of such stochastic processes, it is often useful to study the solutions of the associated deterministic ordinary differential equation that describes the mean behavior of the process. This ODE is

$$\frac{dm}{dt} = -\phi(m), \tag{4}$$

  where $\phi(m)$ is the expectation of $\Phi(., m)$ with respect to the probability distribution of the inputs x (*continuous setting*) or the arithmetic mean (*discrete setting*).
  Here the $k^{th}-$component of ϕ is

$$\phi_k(m) = \sum_{j=1}^{K} h_{kj} \int_{C_j} (x - m_k) f(x) dx, \tag{5}$$

  for the continuous setting or

$$\phi_k(m) = \frac{1}{N} \sum_{j=1}^{K} h_{kj} \sum_{x_i \in C_j} (x_i - m_k), \tag{6}$$

  that can be also written

$$\phi_k(m) = \frac{1}{N} \sum_{i=1}^{N} h_{kc(x_i)} (x_i - m_k), \tag{7}$$

  for the discrete setting.
  The possible limit states of the stochastic process in Equation (2) would have to be solutions of the equation

$$\phi(m) = 0.$$

Then if the zeros of this function were the minima of a function (most often called *energy* function), it would be useful to apply the gradient descent methods.

– The Robbins-Monro algorithm theory which is used when the learning rate decreases under conditions

$$\sum_t \varepsilon(t) = +\infty \text{ and } \sum_t \varepsilon(t)^2 < +\infty. \tag{8}$$

Unfortunately some remarks explain why the original SOM algorithm is difficult to study. Firstly, for dimension $p > 1$, a problem arises: it is not possible to define any absorbing class which could be an organized state. Secondly, although the process $m(t)$ can be written down as a classical stochastic process of Equation (3), one knows since the papers Erwin et al. [1992a,b], that it does not correspond to an energy function, that is it is not a gradient descent algorithm in the *continuous setting*. Finally, it must be emphasized that no demonstration takes into account the variation of the neighborhood function. All the existing results are valid for a fixed size and intensity of the function h.

3 The one-dimensional case

A very particular setting is the one-dimensional case: the inputs belong to $\mathbb{R}$ and the lattice is a one-dimensional array (*a string*). Even though this case is of a poor practical utility, it is interesting because the theoretical analysis can be fully conducted.

3.1 The simplest one-dimensional case

The simplest case was fully studied in the article Cottrell and Fort [1987]. The inputs are supposed to be uniformly distributed in $[0, 1]$, the lattice is a one-dimensional array $\{1, 2, \ldots, K\}$, the learning rate ϵ is a constant smaller than $\frac{1}{2}$, the neighborhood function is a constant step function $h_{kl} = 0$ if $|k - l| > 1$ and 1 otherwise. In that case the process $m(t)$ is a homogeneous Markov Chain with continuous state space. The organization we look for is simply the ordering (ascending or descending) and so is easy to characterize. Let us describe the main steps of the proof.

1. There exists a decreasing functional: the number of badly ordered triplets. But this is not sufficient to prove the convergence, it has to be strictly decreasing with a strictly positive probability.
2. The set of ordered dispositions is an absorbing class, composed of two classes which do not communicate: the increasing sequences class and the decreasing sequences class.
3. One shows that ordering (topology preservation in this special case) takes place after a finite time with a probability which is greater than a positive bound, and that the hitting time of the absorbing class is almost surely finite.

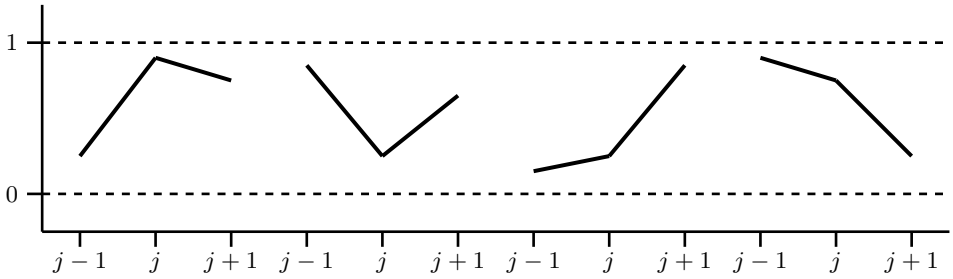

Fig. 1. Four examples of triplets of prototypes (m_{j-1}, m_j, m_{j+1}). For each j, $j-1$ and $j+1$ are its neighbors. The y-axis coordinates are the values of the prototypes that take values in $[0, 1]$. The first two triplets on the left are badly ordered. In the case under study, SOM will order them with a strictly positive probability. The last two triplets (on the right) are well ordered and SOM will never disorder them.

4. Then one shows that the Markov Chain has the Doeblin property: there exists an integer T, and a constant $c > 0$, such that, given that the process starts from any ordered state, and for all set E in $[0, 1]^n$, with positive measure, the probability to enter in E with less than T steps is smaller than $c\,\mathrm{vol}(E)$.
5. This implies that the chain converges in distribution to a monotonous stationary distribution which depends on ϵ (which is a constant in that part).
6. If $\epsilon(t)$ tends towards 0 and satisfies the Robbins-Monro conditions (8), once the state is ordered, the Markov Chain almost surely converges towards a constant (monotonous) solution of an explicit linear system.

So in this very simple case, we could prove the convergence to a unique ordered solution such that

$$m_1(+\infty) < m_2(+\infty) < \ldots < m_K(+\infty),$$

or

$$m_1(+\infty) > m_2(+\infty) > \ldots > m_K(+\infty).$$

Unfortunately, it is not possible to find absorbing classes when the dimension is larger than 1. For example, in dimension 2, with 8 neighbors, if the x- and y-coordinates are ordered, it is possible (with positive probability) to disorder the prototypes as illustrated in Figure 2.

3.2 What we know about the general one-dimensional case

We summarize in this section the essential results that apply to the general one dimension case (with constant neighborhood function and in the continuous setting). References and precise statements can be found in Fort [2006] and Cottrell et al. [1998]. Compared to the previous section, hypothesis on the data distribution and/or the neighborhood function are relaxed.

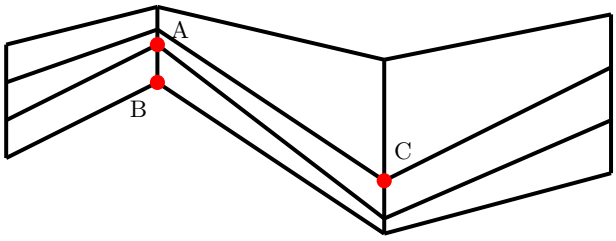

Fig. 2. This figure represents 2-dimensional prototypes (x- and y-axes are not shown but are the standard horizontal and vertical axes) which are linked as their corresponding units on the SOM grid. At this step of the algorithm, the x- and y- coordinates of the prototypes are well ordered. But contrarily to the one-dimensional case, this disposition can be disordered with a positive probability; in an 8-neighbors case, A is C's neighbor, but B is not a neighbor of C. If C is repeatedly the best matching unit, B is never updated, while A becomes closer and closer to C. Finally, the y coordinate of A becomes smaller than that of B and the disposition is disordered.

- The process $m(t)$ is almost surely convergent to a unique stable equilibrium point in a very general case: $\epsilon(t)$ is supposed to satisfy the conditions (8), there are hypotheses on the density f and on the neighborhood function h. Even though these hypotheses are not very restrictive, some important distributions, such as the χ^2 or the power distribution, do not fulfill them.
- For a constant ϵ, the ordering time is almost surely finite (and has a finite exponential moment).
- With the same hypotheses as before to ensure the existence and uniqueness of a stable equilibrium x^*, from any ordered state, for each constant ϵ, there exists an invariant probability measure π^ϵ. When ϵ tends to 0, this measure concentrates on the Dirac measure on x^*.
- With the same hypotheses as before to ensure the existence and uniqueness of a stable equilibrium x^*, from any ordered state, the algorithm converges to this equilibrium provided that $\epsilon(t)$ satisfies the conditions (8).

As the hypotheses are sufficiently general to be satisfied in most cases, one can say that the one-dimensional case is more or less well-known. However nothing is proved neither about the choice of a decreasing function for $\epsilon(t)$ to ensure simultaneously ordering and convergence, nor for the case of decreasing neighborhood function.

4 Multidimensional case

When the data are p-dimensional, one has to distinguish two cases, the continuous setting and the discrete one.

4.1 Continuous setting

In the $p-$dimensional case, we have only partial results proved by Sadeghi in (Sadeghi [2001]). In this paper, the neighborhood function is supposed to have

a finite range, the learning rate ϵ is a constant, the probability density function is positive on an interval (this excludes the discrete case). Then the algorithm weakly converges to a unique probability distribution which depends on ϵ.

Nothing is known about the possible topology preservation properties of this stationary distribution. This is a consequence of the difficulty of defining an absorbing organized state in a multidimensional setting. For example, two results of Flanagan and Fort-Pagès illustrate the complexity of the problem. These two apparently contradictory results hold. For $p = 2$, let us consider the set F^{++} of simultaneously ordered coordinates (respectively x and y coordinates). We then have:

- for a constant ϵ and very general hypotheses on the density f, the hitting time of F^{++} is finite with a positive probability (Flanagan [1996]),
- *but* in the 8-neighbor setting, the exit time is also finite with positive probability (Fort and Pagès [1995]).

4.2 Discrete setting

In this setting, the stochastic process $m(t)$ of Equations (2) and (4) derives from a potential function, which means that it is a gradient descent process associated to the energy. When the neighborhood function does not depend on time, Ritter et al. [1992] have proven that the stochastic process $m(t)$ of Equations (2) and (3) derives from a potential, that is it can be written

$$\begin{aligned} m_k(t+1) &= m_k(t) + \epsilon(t) h_{kc(x)}(t)(x - m_k(t)), \\ &= m_k(t) - \epsilon(t) \Phi_k(x, m(t)), \\ &= m_k(t) - \epsilon(t) \frac{\partial}{\partial m_k} E(x, m(t)), \end{aligned}$$

where $E(x, m)$ is a sample function of $E(m)$ with

$$E(m) = \frac{1}{2N} \sum_{k=1}^{K} \sum_{j=1}^{K} h_{kj} \sum_{x_i \in C_j} \|m_k - x_i\|^2, \tag{9}$$

or in a shorter expression

$$E(m) = \frac{1}{2N} \sum_{i=1}^{N} \sum_{k=1}^{K} h_{kc(x_i)} \|m_k - x_i\|^2. \tag{10}$$

In other words the stochastic process $m(t)$ is a stochastic gradient descent process associated to function $E(m)$. Three interesting remarks can be made:

1. The energy function is a generalization of the *distortion* function (or *intra classes variance* function) associated to the Simple Competitive Learning process (SCL, also known as the Vector Quantization Process/Algorithm),

which is the stochastic version of the deterministic Forgy algorithm. The SCL process is nothing else than the SOM process where the neighborhood function is degenerated, ie when $h_{kl} = 1$ only for $k = l$ and $h_{kl} = 0$ elsewhere. In that case, E reduces to

$$E(m) = \frac{1}{2N} \sum_{i=1}^{N} \|m_{c(x_i)} - x_i\|^2.$$

For that reason, E is called *extended intra-classes variance*.

2. The above result does not ensure the convergence of the process: in fact the gradient of the energy function is not continuous and the general hypotheses used to prove the convergence of the stochastic gradient descent processes are not valid. This comes from the fact that there are discontinuities when crossing the boundaries of the clusters associated to the prototypes, because the neighbors involved in the computation change from a side to another. However this energy gives an interesting insight on the process behavior.
3. In the 0-neighbor setting, the Vector Quantization algorithm converges, since there is no problem with the neighbors and the gradient is continuous. However there are a lot of local minima and the algorithm converges to one of these minima.

5 Deterministic batch SOM

As the possible limit states of the stochastic process (2) would have to be solutions of the ODE equation

$$\phi(m) = 0,$$

it is natural to search how to directly get these solutions. The definition of the batch SOM algorithm can be found in Kohonen [1995, 1999].

From Equation (5), in the continuous setting, the equilibrium m^* must satisfy

$$\forall k \in \mathcal{K}, \qquad \sum_{j=1}^{K} h_{kj} \int_{C_j} (x - m_k^*) f(x) dx.$$

Hence, for the continuous setting, the solution complies with

$$m_k^* = \frac{\sum_{j=1}^{K} h_{kj} \int_{C_j} x f(x) dx}{\sum_{j=1}^{K} h_{kj} \int_{C_j} f(x) dx}.$$

In the discrete setting, the analogous is

$$m_k^* = \frac{\sum_{j=1}^{K} h_{kj} \sum_{x_i \in C_j} x_i}{\sum_{j=1}^{K} h_{kj} |C_j|} = \frac{\sum_{i=1}^{N} h_{kc(x_i)} x_i}{\sum_{i=1}^{N} h_{kc(x_i)}}$$

Thus, the limit prototypes m_k^* have to be the weighted means of all the inputs which belong to the cluster C_k or to its neighboring clusters. The weights are given by the neighborhood function h.

Using this remark, it is possible to derive the definition of the batch algorithm.

$$m_k(t+1) = \frac{\sum_{j=1}^{K} h_{kj}(t) \int_{C_{j(m_k(t))}} x f(x) dx}{\sum_{j=1}^{K} h_{kj}(t) \int_{C_{j(m_k(t))}} f(x) dx}. \tag{11}$$

for the continuous setting, and

$$m_k(t+1) = \frac{\sum_{i=1}^{N} h_{kc^t(x_i)}(t) x_i}{\sum_{i=1}^{N} h_{kc^t(x_i)}(t)} \tag{12}$$

for the discrete case.

This algorithm is deterministic, and one of its advantages is that the limit states of the prototypes depend only on the initial choices. When the neighborhood is reduced to the unit itself, this batch algorithm for the SOM is nothing else than the classical Forgy algorithm (Forgy [1965]) for clustering. Its theoretical basis is solid and a study of the convergence can be found in Cheng [1997]. One can prove (Fort et al. [2001, 2002]) that it is exactly a quasi-Newtonian algorithm associated to the extended distortion (energy) E (see Equation (10)), when the probability to observe a x in the sample which is exactly positioned on the median hyperplanes (e.g., the boundaries of C_k) is equal to zero. This assumption is always true in the continuous setting but it is not relevant in the discrete setting since there is no guarantee that data points never belong to the boundaries which vary along the iterations.

The batch SOM algorithm is the extension of the Forgy algorithm with the introduction of the neighborhood between clusters, in the same way as the on-line SOM algorithm is for the Vector Quantization algorithm. It is not exactly a gradient descent algorithm, but it converges to a minimum of the energy E. Obviously there are many local minima. In conclusion, the relations between these clustering algorithms are summarized in Table 1.

	on-line stochastic	batch deterministic
No neighbor	VQ, SCL, k-means	Forgy, moving centers
With neighbors	SOM	batch SOM

Table 1. Comparison summary.

6 Other algorithms related to SOM

As explained before, the on-line SOM is not a gradient algorithm in the continuous setting (Erwin et al. [1992a,b]). In the discrete setting, there exists an energy function, which is an extended intra-classes variance as in Equation (10), but this function is not continuously differentiable. To overcome these problems, Heskes [1999] proposes to slightly modify the on-line version of the SOM algorithm so it can be seen as a stochastic gradient descent on the same energy function. To do so, he introduces a new hard assignment of the winning unit and a soft version of this assignment.

6.1 Hard assignment in the Heskes's rule

In order to obtain an energy function for the on-line SOM algorithm, Heskes [1999] modifies the rule for computing the best matching unit (BMU). In his setting, Equation (1) becomes

$$c^t(x) = \arg \min_{k \in \{1,\dots,K\}} \sum_{j=1}^{K} h_{kj}(t) \|x - m_k(t)\|^2 \tag{13}$$

The energy function considered here is

$$E(m) = \frac{1}{2} \sum_{j=1}^{K} \sum_{k=1}^{K} h_{kj}(t) \int_{x \in C_j(m)} \|x - m_k(t)\|^2 f(x) dx, \tag{14}$$

where $C_j(m)$ is the cluster (Voronoï cell) associated to the j-th prototype. The regularity properties of the energy function and of its gradient are summarized in Table 2, as discussed in Heskes [1999].

	Discrete setting	Continuous setting
Kohonen rule for computing BMU	Energy : discontinuous (but finite on V) Gradient : discontinuous (infinite on V)	Energy : continuous Gradient : discontinuous
Heskes rule for computing BMU	Energy : continuous Gradient : discontinuous (finite on V)	Energy : continuous Gradient : continuous

Table 2. Smoothness of the energy function

6.2 Soft Topographic Mapping (STM)

The original SOM algorithm is based on a hard winner assignment. Generalizations based on soft assignments were derived in Heskes [1999] and Graepel et al. [1998]. First, let us remark that the energy function in the discrete case can also be written as

$$E(m,c) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{N} c_{ik}\sum_{j=1}^{K} h_{kj}(t)\|m_j(t) - x_i\|^2$$

where c_{ik} is equal to 1 if x_i belongs to cluster k and zero otherwise. This crisp assignment may be smoothed by considering $c_{ik} \geq 0$ such that $\sum_{k=1}^{K} c_{ik} = 1$. The soft assignments may be viewed as the probabilities of input x_i to belong to class k.

Since the optimization of the energy function with gradient descent-like algorithms would get stuck into local minima, the problem is transformed into a deterministic annealing scheme. The energy function is smoothed by adding an entropy term and transforming it into a "free energy" cost function, parameterized by a parameter β :

$$F(m,c,\beta) = E(m,c) - \frac{1}{\beta}S(c) \ ,$$

where $S(c)$ is the entropy term associated to the full energy. For low values of β, only one global minimum remains and may be easily determined by gradient descent or EM schemes. For $\beta \to +\infty$, the free energy has exactly the same expression as the original energy function.
When using deterministic annealing, one begins by computing the minimum of the free energy at low values of β and then attempts to compute the minimum for higher values of β (β may be chosen to grow exponentially), until the global minimum of the free energy for $\beta \to +\infty$ is equal to the global minimum of the original energy function.
For a fixed value of β, the minimization of the free energy leads to iterating over two steps given by Equations (15) and (16), in batch version, and very similar to the original SOM (the neighborhood function h is not varied during the optimization process) :

$$\mathbb{P}(x_i \in C_k) = \frac{\exp(-\beta e_{ik})}{\sum_{j=1}^{K}\exp(-\beta e_{ij})}, \tag{15}$$

where $e_{ik} = \frac{1}{2}\sum_{j=1}^{K} h_{jk}(t)\|x_i - m_j(t)\|^2$ and

$$m_k(t) = \frac{\sum_{i=1}^{N} x_i \sum_{j=1}^{K} h_{jk}(t)\mathbb{P}(x_i \in C_j)}{\sum_{i=1}^{N}\sum_{j=1}^{K} h_{jk}(t)\mathbb{P}(x_i \in C_j)} \tag{16}$$

The updated prototypes are written as weighted averages over the data vectors. For $\beta \to +\infty$, the classical batch SOM is retrieved.

6.3 Probabilistic views on the SOM

Several attempts have been made in order to recast the SOM algorithm (and its variants) into a probabilistic framework, namely the general idea of mixture models (see e.g. McLachlan and Peel [2004]). The central idea of those approaches is to constrain a mixture of Gaussian distributions in a way that mimic the SOM grid. Due to the heuristic nature of the SOM, the resulting models depart quite significantly from the SOM algorithms and/or from standard mixture models. We describe below three important variants. Other variants are listed in e.g. Verbeek et al. [2005].

SOM and regularized EM One of the first attempt in this direction can be found in Heskes [2001]. Based on his work on energy functions for the SOM, Heskes shows in this paper that the batch SOM can be seen as a form of regularized Expectation Maximization (EM) algorithm[1].

As mentioned above, the starting point of the this analysis consists in introducing an isotropic Gaussian mixture with K components. The multivariate Gaussian distributions share a single precision parameter β, with the covariance matrix $\frac{1}{\beta}\mathbf{I}$, and are centered on the prototypes.

However, up to some constant terms, the opposite of the log likelihood of such a mixture corresponds to the k-means quantization error. And therefore, maximizing the likelihood does not provide any topology preservation. Thus Heskes introduces a regularization term which penalizes prototypes that do not respect the prior structure (the term does not depend directly on the data points), see Heskes [2001] for details. Then Heskes shows that applying the EM principle to the obtained regularized (log)likelihood leads to an algorithm that resembles the batch SOM one.

This interpretation has very interesting consequences, explored in the paper. It is easy for instance to leverage the probabilistic framework to handle missing values in a principled (non heuristic) way. It is also easy to use other mixtures e.g. for non numerical data (such as count data). However, the regularization itself is rather ad hoc (it cannot be easily interpreted as a prior distribution on the parameters, for instance). In addition, the final algorithm is significantly different from the batch SOM. Indeed, as in the case of the STM, crisp assignments are replaced by probabilistic ones (the crispness of the assignments is controlled by the precision parameter β). In addition, as in STM, the neighborhood function is fixed (as it is the core of the regularization term). To our knowledge, the practical consequences of those differences have not been studied in detail on real world data. While one can argue that β can be increased progressively and at the same time, one can modify the neighborhood function during the EM algorithm, this might also have consequences that remain untested.

SOM and variational EM Another take at this probabilistic interpretation can be found in Verbeek et al. [2005]. As in Heskes [2001] the first step consists

[1] EM is the standard algorithm for mixture models.

in assuming a standard mixture model (e.g. a K components Gaussian isotropic mixture for multivariate data). Then the paper leverages the variational principle (see e.g. Jordan et al. [1999]).

In summary, the variational principle is based on introducing an arbitrary distribution q on the latent (hidden) variables Z of the problem under study. In a standard mixture model, the hidden variables are the assignment ones, which map each data point to a component of the mixture (a cluster in the standard clustering language). One can show that the integrated log likelihood of a mixture model with Θ as parameters, $\log p(X|\Theta)$, is equal to the sum of three components: the complete likelihood (knowing both the data points X and the hidden variables Z) integrated over the hidden variables with respect to q, $\mathbb{E}_q \log p(X, Z|\Theta)$, the entropy of q, $H(q)$, and the Kullback-Leibler divergence, $KL(q|p(Z|X, \Theta))$, between q and the posterior distribution of the hidden variables knowing the data points $p(Z|X, \Theta)$. This equality allows one to derive the EM algorithm when the posterior distribution of the hidden variables knowing the data points can be calculated exactly. The variational approach consists in replacing this distribution by a simpler one when it cannot be calculated.

In standard mixture models (such as the multivariate Gaussian mixture), the variational approach is not useful as the posterior distribution of the hidden variables can be calculated. However Verbeek et al. [2005] propose nevertheless to use the variational approach as a way to enforce regularity in the mixture model. Rather than allowing $p(Z|X, \Theta)$ to take an arbitrary form, they constrain it to a subset of probability distributions on the hidden variables that fulfill topological constraint corresponding to the prior structure of the SOM. See Verbeek et al. [2005] for details.

This solution shares most of the advantages of the older proposal in Heskes [2001], with the added value of being based on a more general principle that can be applied to any mixture model (in practice, Heskes [2001] makes sense only for the exponential family). In addition, Verbeek et al. [2005] study the effects of shrinking the neighborhood function during training and conclude that it improves the quality of the solutions. Notice that, in Verbeek et al. [2005], the shared precision of the Gaussian distributions (β) is not a meta-parameter as in Heskes [2001] but a regular parameter that is learned from the data.

The Generative Topographic Mapping The Generative Topographic Mapping (GTM, Bishop et al. [1998]) is frequently presented as a probabilistic version of the SOM. It is rather a mixture model inspired by the SOM rather than an adaptation. Indeed the aim of the GTM designers was not to recover a learning algorithm close to a SOM variant, but rather to introduce a mixture model that enforce topology preservation.

The GTM is based on uniform prior distribution on a fixed grid which is mapped via an explicit smooth nonlinear mapping to the data space (with some added isotropic Gaussian noise). It can be seen as a constrained Gaussian mixture, but with yet another point of view compared to Heskes [2001] and Verbeek et al. [2005]. In Heskes [2001], the constraint is enforced by a regularization term

on the data space distribution while in Verbeek et al. [2005] the constraint is induced at the latent variable level (via approximating $p(Z|X, \Theta)$ by a smooth distribution). In the GTM the constraint is induced on the data space distribution because it is computed via a smooth mapping. In other words, the centers of the Gaussian distributions are not freely chosen but rather obtained by mapping a fixed grid to the data space via the nonlinear mapping.

The nonlinear mapping is in principle arbitrary and can therefore implement various type of regularity (i.e. topology constraints). The use of Gaussian kernels lead to constraints that are quite similar to the SOM constraints. Notice that those Gaussian kernels are not to be confused with the isotropic Gaussian distributions used in the data space (the same confusion could arise in Verbeek et al. [2005] where Gaussian kernels can be used to specify the constraints on $p(Z|X, \Theta)$).

Once the model has been specify (by choosing the nonlinear mapping), its parameters are estimated via an EM algorithm. The obtained algorithm is quite different from the SOM (see Heskes [2001] for details), at least in its natural formulation. However the detailed analysis contained in Heskes [2001] shows that the GTM can be reformulated in a way that is close to the batch SOM with probabilistic assignments (as in e.g. the STM). Once again, however, this is not exactly the same algorithm. In practice, the results on real world data can be quite different. Also, as all the probabilistic variants discussed in this section, the GTM benefits from the probabilistic setting that enables principled missing data analysis as well as easy extensions to the exponential family of distributions in order to deal with non numerical data.

7 Non numerical data

When the data are not numerical, the SOM algorithm has to be adapted. See for example Kohonen [1985], Kohonen [1996], Kaski et al. [1998a], Kohohen and Somervuo [1998], Kohonen [2001], Kohonen and Somervuo [2002], Kohonen [2013], Kohonen [2014], Cottrell et al. [2012], where some of these adaptations are presented. Here we deal with categorical data collected in surveys and with abstract data which are known only by a dissimilarity matrix or a kernel matrix.

7.1 Contingency table or complete disjunctive table

Surveys collect answers of the surveyed individuals who have to choose an answer to several questions among a finite set of possible answers. The data can consist in

- a simple contingency table, where there are only two questions, and where the entries are the numbers of individuals who choose a given pair of categories,
- a Burt table, that is a full contingency table between all the pairs of categories of all the questions,

– a complete disjunctive table that contains the answers of all the individuals, coded in 0/1 against dummy variables which represent all the categories of all the questions.

In all these settings, the data consist in a positive integer-valued matrix, which can be seen as a large "contingency table". In classical data analysis, one uses Multiple Correspondence Analysis (MCA) that are designed to deal with these tables. MCA is nothing else than two simultaneous weighted Principal Component Analysis (PCA) of the table and of its transposed, using the χ^2 distance instead of the Euclidean distance. To use a SOM algorithm with such tables, it is therefore sufficient to apply a transformation to the data, in order to take into account the χ^2 distance and the weighting, in the same way that it is defined to use Multiple Correspondence Analysis. After transforming the data, two coupled SOM using the rows and the columns of the transformed table can thus be trained. In the final map, related categories belong to the same cluster or to neighboring clusters. The reader interested by a detailed explanation of the algorithm can refer to Bourgeois et al. [2015a] . More details and real-world examples can also be found in Cottrell et al. [2004], Cottrell and Letrémy [2005]. Notice that the transformed tables are numerical data tables, and so there is no particular theoretical results to comment on. All the results that we presented for numerical data still hold.

7.2 Dissimilarity Data

In some cases, complex data such as graphs (social networks) or sequences (DNA sequences) are described through relational measures of resemblance or dissemblance, such as kernels or dissimilarity matrices. For these general situations, several extensions of the original algorithm, both in on-line and batch versions, were proposed during the last two decades. A detailed review of these algorithms is available in Rossi [2014].

More precisely, these extensions consider the case where the data are valued in an arbitrary space $\mathcal{X}$, which is not necessarily Euclidean. The observations are described either by a pairwise dissimilarity $\mathbf{D} = (\delta(x_i, x_j))_{i,j=1,\ldots,N}$, or by a kernel matrix $\mathbf{K} = (K(x_i, x_j))_{i,j=1,\ldots,N}$[2]. The kernel matrix $\mathbf{K}$ naturally induces an Euclidean distance matrix, but the dissimilarity matrix $\mathbf{D}$ may not necessarily be transformed into a kernel matrix.

The first class of algorithms designed for handling relational data is based on the median principle (*median SOM*): prototypes are forced to be equal to an observation, or to a fixed number of observations. Hence, optimal prototypes

[2] A kernel is a particular case of symmetric similarity such that K is a symmetric matrix, semi-definite positive with $K(x_i, x_i) = 0$ and satisfies the following positive constraint

$$\forall\, M > 0,\ \forall\, (x_i)_{i=1,\ldots,M} \in \mathcal{X},\ \forall\, (\alpha_i)_{i=1,\ldots,M}, \qquad \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \geq 0.$$

are computed by searching through $(x_i)_{i=1,\dots,N}$, instead of $\mathcal{X}$, as in Kohohen and Somervuo [1998], Kohonen and Somervuo [2002], El Golli et al. [2004a] and El Golli et al. [2004b]. The original steps of the algorithm are thus transformed in a discrete optimization scheme, which is performed in batch mode:

1. Equation (1) is replaced by the affectation of *all* data to their best matching units: $c(x_i) = \arg\ \min_{k=1,\dots,K} \delta(x_i, m_k(t))$;
2. Equation (2) is replaced by the update of all prototypes within the dataset $(x_i)_{i=1,\dots,N} : m_k(t) = \min_{x_i\ :i=1,\dots,N} \sum_{j=1}^{N} h_{c(x_i)j}(t)\delta(x_i, x_j)$.

Since the algorithm explores a finite set when updating the prototypes, it is necessarily convergent to a local minimum of the energy function. However, this class of algorithms exhibits strong limitations, mainly due to the restriction of the prototypes to the dataset, in particular, a large computational cost (despite efficient implementations such as in Conan-Guez et al. [2006]) and no interpolation effect which yields to a deterioration of the quality of the map organization.

The *second class* of algorithms, *kernel SOM and relational/dissimilarity SOM*, rely on expressing prototypes as convex combinations of the input data. Although these convex combinations do not usually have sense in $\mathcal{X}$ (consider, for instance, that input data are various texts), a convenient embedding in an Euclidean or a pseudo-Euclidean space gives a sound theoretical framework and gives sense to linear combinations of inputs.

For *kernel SOM*, it is enough to use the kernel trick as given by Aronszajn [1950] which prove that there exists a Hilbert space $\mathcal{H}$, also called feature space, and a mapping $\psi : \mathcal{X} \to \mathcal{H}$, called feature map, such that $K(x, x') = \langle \psi(x), \psi(x') \rangle_{\mathcal{H}}$. In the case where data are described by a *symmetric dissimilarity measure*, they may be embedded in a pseudo-Euclidean space $\psi : x \in \mathcal{X} \to \psi(x) = (\psi^+(x), \psi^-(x)) \in \mathcal{E}$, as suggested in Goldfarb [1984]. $\mathcal{E}$ may be written as the direct decomposition of two Euclidean spaces, $\mathcal{E}_+$ and $\mathcal{E}_-$, with a non-degenerate and indefinite inner product defined as

$$\langle \psi(x), \psi(y) \rangle_{\mathcal{E}} = \langle \psi^+(x), \psi^+(u) \rangle_{\mathcal{E}^+} - \langle \psi^-(x), \psi^-(u) \rangle_{\mathcal{E}^-}$$

The distance naturally induced by the pseudo-Euclidean inner product is not necessarily positive.

For both kernel and relational/dissimilarity SOM, the input data are embedded in $\mathcal{H}$ or $\mathcal{E}$ and prototypes are expressed as convex combinations of the images of the data by the feature maps. For example, in the kernel case,

$$m_k(t) = \sum_{i=1}^{N} \gamma_{ki}^t \psi(x_i) \text{ , with } \gamma_{ki}^t \geq 0 \text{ and } \sum_i \gamma_{ki}^t = 1.$$

The above writing of the prototypes allows the computation of the distance from an input x_i to a prototype $m_k(t)$ in terms of the coefficients γ_{ki}^t and the kernel/dissimilarity matrix only. For kernel SOM, one has

$$\|\psi(x_i) - m_k(t)\|^2 = \left(\gamma_k^t\right)^T \mathbf{K}\gamma_k^t - 2\mathbf{K}_i\gamma_k^t + \mathbf{K}_{ii} \text{ ,}$$

where $\mathbf{K}_i$ is the ith row of $\mathbf{K}$ and $\left(\gamma_k^t\right)^T = \left(\gamma_{k,1}^t, ..., \gamma_{k,N}^t\right)$. For relational/dissimilarity SOM, one obtains a similar expression

$$\|\psi(x_i) - m_k(t)\|^2 = \mathbf{D}_i\gamma_k^t - \frac{1}{2}\left(\gamma_k^t\right)^T \mathbf{D}\gamma_k^t \ .$$

The first step of the algorithm, finding the best matching unit of an observation, as introduced in Equation (1), can thus be directly generalized to kernels and dissimilarities, both for on-line and batch settings.

In the batch framework, the updates of the prototypes are identical to the original algorithm (see Equation (12)), by simply noting that only the coefficients of the x_i's (or of their images by the feature maps) are updated :

$$m_k(t+1) = \sum_{i=1}^{N} \frac{h_{kc^t(x_i)}(t)}{\sum_{j=1}^{N} h_{kc^t(x_j)}(t)}\psi(x_i) \Leftrightarrow \gamma_{ki}^{t+1} = \frac{h_{kc^t(x_i)}(t)}{\sum_{j=1}^{N} h_{kc^t(x_j)}(t)} \tag{17}$$

This step is the same, both for batch kernel SOM, Villa and Rossi [2007], and for batch relational SOM, Hammer et al. [2007].

In the on-line framework, updating the prototypes is similar to the original algorithm, as in Equation (2). Here also, the update rule concerns the coefficients γ_{ki}^t only, and the linear combination of them remains convex :

$$\gamma_k^{t+1} = \gamma_k^t + \varepsilon(t)h_{kc^t(x_i)}(t)\left(\mathbf{1}_i - \gamma_k^t\right) \ , \tag{18}$$

where x_i is the current observation and $\mathbf{1}_i$ is a vector in $\mathbb{R}^N$, with a single non-null coefficient, equal to 1, on the i-th position. As previously, this step is identical for on-line kernel SOM, Mac Donald and Fyfe [2000], and for on-line relational SOM, Olteanu and Villa-Vialaneix [2015].

In the case where the dissimilarity is the squared distance induced by the kernel, kernel SOM and relational SOM are strictly equivalent. Moreover, in this case, they are also fully equivalent to the original SOM algorithm for numerical data in the feature (implicit) Euclidean space induced by the dissimilarity or the kernel, as long as the prototypes are initialized in the convex hull of the input data. The latter assertion induces that the theoretical limitations of the original algorithm also exist for the general kernel/relational versions. Furthermore, these may worsen for the relational version since the non-positivity of the dissimilarity measure adds numerical instability when using a gradient-descent like scheme for updating the prototypes.

The *third class* of algorithms uses the *soft topographic maps* setting introduced in section 6.2. Indeed, in the algorithm described in equations (15) and (16), the soft assignments depend on the distances between input data and prototypes only, while prototypes update consists in making an update if the coefficients of the input data. Using a mean-field approach and similarly to the previous framework for kernel and dissimilarity/relational SOM, Graepel et al. [1998] obtain the extensions of soft topographic mapping (STM) algorithm for

kernels and dissimilarities. The updates for the prototype coefficients are then expressed as

$$\gamma_{ki}(t+1) = \frac{\sum_{j=1}^{K} h_{jk}(t)\mathbb{P}(x_i \in A_j)}{\sum_{l=1}^{N}\sum_{j=1}^{K} h_{jk}(t)\mathbb{P}(x_l \in A_j)}, \tag{19}$$

where $m_k(t) = \sum_{i=1}^{N} \gamma_{ki}^t \psi(x_i)$ and ψ is the feature map.

8 Stochasticity of the Kohonen maps for the on-line algorithm

Starting from a given initialization and a given size of the map, different runs of the on-line stochastic SOM algorithm provide different resulting maps. On the contrary, the batch version of the algorithm is a deterministic algorithm with always provides the same results for a given initialization. For this reason, the batch SOM algorithm is often preferred over the stochastic one because its results are reproducible. However, this hides the fact that all the pairs of observations which are associated in a given cluster do not have the same significance. More precisely, interpreting a SOM result, we can use the fact that close input data belong to close clusters, i.e. their best matching units are identical or adjacent. But if two given observations are classified in the same or in neighboring units of the map, then they may not be close in the input space. This drawback comes from the fact that there is not perfect matching between a a multidimensional space and a one- or two-dimensional map.

More precisely, given a pair of observations (data), $\{x_i, x_j\}$, three cases can be distinguished, depending on the way their respective mapping on the map can be described:

- *significant association*: the pair is classified in the same cluster or in neighboring clusters because x_i and x_j are close in the input space. The observations are said to attract each other;
- *significant non-association*: the pair is never classified in neighboring clusters and x_i and x_j are remote in the input space. The observations are said to repulse each other;
- *fickle pair*: the pair is sometimes classified in the same cluster or in neighboring clusters but x_i and x_j are not close in the input space: their proximity on the map is due to randomness.

The stochasticity of the on-line SOM results can be used to precisely qualify every pairs of observations by performing several runs of the algorithm. The question is addressed in a bootstrap framework in de Bodt et al. [2002] and used for text mining applications in Bourgeois et al. [2015a,b]. The idea is simple: since the on-line SOM algorithm is stochastic, its repetitive use may allow to identify the pairs of data in each case.

More precisely, if L is the number of different and independent runs of the on-line SOM algorithm and if $Y_{i,j}$ denotes the number of times x_i and x_j are

neighbors on the resulting map in the L runs, a *stability index* can be defined: for the pair (x_i, x_j), this index is equal to:

$$\mathcal{M}_{i,j} = \frac{Y_{i,j}}{L}.$$

Using an approximation of the binomial distribution that would hold if the data were neighbors by chance in a pure random way, and a test level of 5%, for a K-units map, the following quantities are introduced

$$A = \frac{9}{K} \text{ and } B = 1.96\sqrt{\frac{9}{KL}(1 - \frac{9}{K})}. \tag{20}$$

These values give the following decision rule to qualify every pair $\{x_i, x_j\}$:

- if $\mathcal{M}_{i,j} > A+B$, the association between the two observations is significantly frequent;
- if $A - B \leq \mathcal{M}_{i,j} \leq A + B$, the association between the two observations is due to randomness. $\{x_i, x_j\}$ is called a fickle pair;
- if $\mathcal{M}_{i,j} < A - B$, the non-association between the two observations is significantly frequent.

In de Bodt et al. [2002], the method is used in order to qualify the stability and the reliability of the global Kohonen map, while both other papers (Bourgeois et al. [2015a] and Bourgeois et al. [2015b]) study the fickle data pairs for themselves. In these last work, the authors also introduce the notion of *fickle word* which is defined as an observation which belongs to a huge number of *fickle pairs* by choosing a threshold.

These fickle pairs and fickle words can be useful in various way: first, fickle pairs can be used to obtain more robust maps, by distinguishing stable neighboring and non neighboring pairs from fickle pairs. Also, once identified, fickle words can be removed from further studies and representations: for instance, Factorial Analysis visualization is improved. In a text mining setting, Bourgeois et al. [2015b] have shown that a graph of co-occurrences between words can be simplified by removing fickle words and Bourgeois et al. [2015a] have used the fickle words for interpretation: they have shown that the fickle words form a lexicon shared between the studied texts.

9 Conclusion

We have reviewed some of the variants of the SOM, for numerical and non numerical data, in their stochastic (on-line) and batch versions. Even if a lot of theoretical properties are not rigorously proven, the SOM algorithms are very useful tools for data analysis in different contexts. Since the Heskes's variants of SOM have a more solid theoretical background, SOM can appear as an easy-to-develop approximation of these well-founded algorithms. This remark should ease the concern that one might have about it.

On a practical point of view, SOM is used as a statistical tool which has to be combined with other techniques, for the purpose of visualization, of vector quantization acceleration, graph construction, etc. Moreover, in a big data context, SOM-derived algorithms seem to have a great future ahead since the computational complexity of SOM is low (proportional to the number of data). In addition, it is always possible to train the model with a sample randomly extracted from the database and then to continue the training in order to adapt the prototypes and the map to the whole database. As most of the stochastic algorithm, SOM is particularly well suited for stream data (see Hammer and Hasenfuss [2010] which proposed a "patch SOM" to handle this kind of data). Finally, it would also be interesting to have a look at the robust associations revealed by SOM, to improve the representation and the interpretation of too verbose and complex information.

References

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

N. Bourgeois, M. Cottrell, B. Deruelle, S. Lamassé, and P. Letrémy. How to improve robustness in kohonen maps and display additional information in factorial analysis: Application to text minin. *Neurocomputing*, 147:120–135, 2015a.

N. Bourgeois, M. Cottrell, S. Lamassé, and M. Olteanu. Search for meaning through the study of co-occurrences in texts. In I. Rojas, G. Joya, and A. Catala, editors, *Advances in Computational Intelligence, Proceedings of IWANN 2015, Part II*, volume 9095 of *LNCS*, pages 578–591. Springer, 2015b.

Y. Cheng. Convergence and ordering of kohonen's batch map. *Neural Computation*, 9: 1667–1676, 1997.

B. Conan-Guez, F. Rossi, and A. El Golli. Fast algorithm and implementation of dissimilarity self-organizing maps. *Neural Networks*, 19(6-7):855–863, 2006.

M. Cottrell and J.C. Fort. Étude d'un processus d'auto-organisation. *Annales de l'IHP, section B*, 23(1):1–20, 1987.

M. Cottrell and P. Letrémy. How to use the Kohonen algorithm to simultaneously analyse individuals in a survey. *Neurocomputing*, 63:193–207, 2005.

M. Cottrell, J.C. Fort, and G. Pagès. Theoretical aspects of the SOM algorithm. *Neurocomputing*, 21:119–138, 1998.

M. Cottrell, S. Ibbou, and P. Letrémy. Som-based algorithms for qualitative variables. *Neural Networks*, 17:1149–1167, 2004.

M. Cottrell, M. Olteanu, F. Rossi, J. Rynkiewicz, and N. Villa-Vialaneix. Neural networks for complex data. *Künstliche Intelligenz*, 26(2):1–8, 2012. doi: 10.1007/s13218-012-0207-2.

E. de Bodt, M. Cottrell, and M. Verleisen. Statistical tools to assess the reliability of self-organizing maps. *Neural Networks*, 15(8-9):967–978, 2002. doi: 10.1016/S0893-6080(02)00071-0.

Aïcha El Golli, Brieuc Conan-Guez, and Fabrice Rossi. Self organizing map and symbolic data. *Journal of Symbolic Data Analysis*, 2(1), 11 2004a.

Aïcha El Golli, Brieuc Conan-Guez, and Fabrice Rossi. A self organizing map for dissimilarity data. In D. Banks, L. House, F. R. McMorris, P. Arabie, and W. Gaul, editors, *Classification, Clustering, and Data Mining Applications (Proceedings of IFCS 2004)*, pages 61–68, Chicago, Illinois (USA), 7 2004b. IFCS, Springer.

E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, 67(1):47–55, 1992a.

E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: stationnary states, metastability and convergence rate. *Biological Cybernetics*, 67(1):35–45, 1992b.

J.A. Flanagan. Self-organisation in kohonen's som. *Neural Networks*, 6(7):1185–1197, 1996.

E.W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, (21):768–769, 1965.

J.-C. Fort, M. Cottrell, and P. Letrémy. Stochastic on-line algorithm versus batch algorithm for quantization and self organizing maps. In *Neural Networks for Signal Processing XI, 2001, Proceedings of the 2001 IEEE Signal Processing Sociéty Workshop*, pages 43–52, North Falmouth, MA, USA, 2001. IEEE.

J.-C. Fort, P. Letrémy, and M. Cottrell. Advantages and drawbacks of the batch kohonen algorithm. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2002)*, pages 223–230, Bruges, Belgium, 2002. d-side publications.

J.C. Fort. SOM's mathematics. *Neural Networks*, 19(6-7):812–816, 2006.

J.C. Fort and G. Pagès. About the kohonen algorithm: strong or weak self-organisation. *Neural Networks*, 9(5):773–785, 1995.

L. Goldfarb. A unified approach to pattern recognition. *Pattern Recognition*, 17(5): 575–582, 1984. doi: 10.1016/0031-3203(84)90056-6.

T. Graepel, M. Burger, and K. Obermayer. Self-organizing maps: generalizations and new optimization techniques. *Neurocomputing*, 21:173–190, 1998.

B. Hammer and A. Hasenfuss. Topographic mapping of large dissimilarity data sets. *Neural Computation*, 22(9):2229–2284, September 2010.

B. Hammer, A. Hasenfuss, F. Rossi, and M. Strickert. Topographic processing of relational data. In Bielefeld University Neuroinformatics Group, editor, *Proceedings of the 6th Workshop on Self-Organizing Maps (WSOM 07)*, Bielefeld, Germany, September 2007.

T. Heskes. Energy functions for self-organizing maps. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 303–315. Elsevier, Amsterdam, 1999. URL `http://www.snn.ru.nl/reports/Heskes.wsom.ps.gz`.

Tom Heskes. Self-organizing maps, vector quantization, and mixture modeling. *IEEE Transactions on Neural Networks*, 12(6):1299–1305, November 2001.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233, 1999.

S. Kaski, T. Honkela, K. Lagus, and T. Kohonen. Websom - self-organizing maps of document collections. *Neurocomputing*, 21(1):101–117, 1998a.

Samuel Kaski, Kangasb Jari, and Teuvo Kohonen. Bibliography of self-organizing map (SOM) papers: 1981–1997. *Neural Computing Surveys*, 1(3&4):1–176, 1998b.

T. Kohohen and P.J. Somervuo. Self-organizing maps of symbol strings. *Neurocomputing*, 21:19–30, 1998.

T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybern.*, 43:59–69, 1982a.

T. Kohonen. Analysis of a simple self-organizing process. *Biol. Cybern.*, 44:135–140, 1982b.

T. Kohonen. Median strings. *Pattern Recognition Letters*, 3:309–313, 1985.
T. Kohonen. *Self-organization and associative memory*. Springer, 1989.
T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Science*. Springer, 1995.
T. Kohonen. Self-organizing maps of symbol strings. Technical report a42, Laboratory of computer and information science, Helsinki University of technoligy, Finland, 1996.
T. Kohonen. Comparison of som point densities based on different criteria. *Neural Computation*, 11:2081–2095, 1999.
T. Kohonen. *Self-Organizing Maps, 3rd Edition*, volume 30. Springer, Berlin, Heidelberg, New York, 2001.
T. Kohonen. Essentials of self-organizing map. *Neural Networks*, 37:52–65, 2013.
T. Kohonen. *MATLAB Implementations and Applications of the Self-Organizing Map*. Unigrafia Oy, Helsinki, Finland, 2014.
T. Kohonen and P.J. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15(8):945–952, 2002.
D. Mac Donald and C. Fyfe. The kernel self organising map. In *Proceedings of 4th International Conference on knowledge-based Intelligence Engineering Systems and Applied Technologies*, pages 317–320, 2000.
Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.
Merja Oja, Samuel Kaski, and Teuvo Kohonen. Bibliography of self-organizing map (SOM) papers: 1998–2001 addendum. *Neural Computing Surveys*, 3:1–156, 2003.
M. Olteanu and N. Villa-Vialaneix. On-line relational and multiple relational SOM. *Neurocomputing*, 147:15–30, 2015. doi: 10.1016/j.neucom.2013.11.047.
David Pollard et al. Strong consistency of k-means clustering. *The Annals of Statistics*, 9(1):135–140, 1981.
H. Ritter and K. Schulten. On the stationary state of kohonen's self-organizing sensory mapping. *Biol. Cybern.*, 54:99–106, 1986.
H. Ritter, T. Martinetz, and K. Schulten. *Neural computation and self-Organizing Maps, an Introduction*. Addison-Wesley, 1992.
F. Rossi. How many dissimilarity/kernel self organizing map variants do we need? In T. Villmann, F.M. Schleif, M. Kaden, and M. Lange, editors, *Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2014)*, volume 295 of *Advances in Intelligent Systems and Computing*, pages 3–23, Mittweida, Germany, 2014. Springer Verlag, Berlin, Heidelberg. doi: 10.1007/978-3-319-07695-9_1.
A. Sadeghi. Convergence in distribution of the multi-dimensional kohonen algorithm. *J. of Appl. Probab.*, 38(1):136–151, 2001.
Jakob J. Verbeek, Nikos Vlassis, and Ben J. A. Kröse. Self-organizing mixture models. *Neurocomputing*, 63:99–123, January 2005.
N. Villa and F. Rossi. A comparison between dissimilarity SOM and kernel SOM for clustering the vertices of a graph. In *6th International Workshop on Self-Organizing Maps (WSOM 2007)*, Bielefeld, Germany, 2007. Neuroinformatics Group, Bielefeld University. ISBN 978-3-00-022473-7. doi: 10.2390/biecoll-wsom2007-139.